Lessons from a Decade of Replications at the *Quarterly Journal of Political Science*

Nicholas Eubank, *Stanford Graduate School of Business*

ABSTRACT To allow researchers to investigate not only whether a paper's methods are theoretically sound but also whether they have been properly implemented and are robust to alternative specifications, it is necessary that published papers be accompanied by their underlying data and code. This article describes experiences and lessons learned at the *Quarterly Journal of Political Science* since it began requiring authors to provide this type of replication code in 2005. It finds that of the 24 empirical papers subjected to in-house replication review since September 2012, only four packages did not require any modifications. Most troubling, 14 packages (58%) had results in the paper that differed from those generated by the author's own code. Based on these experiences, this article presents a set of guidelines for authors and journals for improving the reliability and usability of replication packages.

The success of science depends critically on the ability of peers to interrogate published research in an effort not only to confirm its validity but also to extend its scope and probe its limitations. Yet, as social science has become increasingly dependent on computational analyses, traditional means of ensuring the accessibility of research—such as peer review of written academic publications—are no longer sufficient. To truly ensure the integrity of academic research moving forward, it is imperative that published papers be accompanied by the code used to generate results. This will allow other researchers to investigate not only whether a paper's methods are theoretically sound but also whether they have been properly implemented and are robust to alternative specifications.

Since its inception in 2005, the *Quarterly Journal of Political Science* (*QJPS*) has sought to encourage this type of transparency by requiring that all submissions be accompanied by a replication package consisting of data and code for generating results. These packages are then made available with the paper on the *QJPS* website. In addition, all replication packages are subject to internal review by the *QJPS* before publication. This internal review, conducted by a graduate student employed by the *QJPS*, includes ensuring that the code executes smoothly, results from the paper can be easily located, and results generated by the replication package match those in the paper.

This policy is motivated by the belief that publication of replication materials serves at least three important academic purposes. First, it directly ensures the integrity of results published in the *QJPS*. Although the in-house screening process constitutes a minimum bar for replication, it nevertheless has identified a remarkable number of problems in papers. From September 2012 to November 2015, for example, 14 of the 24 empirical papers subject to in-house review were found to have discrepancies between the results generated by authors' own code and those in their written manuscripts.

Second, by emphasizing the need for transparent and easy-to-interpret code, the *QJPS* expects to lower the costs associated with other scholars interrogating the results of existing papers. This increases the probability that other scholars will examine the code for published papers, potentially identifying errors or issues of robustness if they exist. In addition, whereas not all code is likely to be examined in detail, it is the expectation of the *QJPS* that this transparency will motivate submitting authors to be especially cautious in their coding and robustness checks, thereby preventing errors before they occur.

Third, publication of transparent replication packages facilitates research that builds on previous work. Many papers published in the *QJPS* represent methodological innovations, and by making the code that underlies those innovations publicly

Nicholas Eubank *is a PhD candidate in political economy at Stanford Graduate School of Business. He was replication assistant for the* Quarterly Journal of Political Science *from September 2012 to November 2015. He can be reached at nicholaseubank@stanford.edu.*

An earlier version of this article appeared in The Political Methodologist, *available at http://thepoliticalmethodologist.com/2014/12/09/a-decade-of-replications-lessons-from-the-quarterly-journal-of-political-science.*

doi:10.1017/S1049096516000196

accessible, *QJPS* expects to lower the cost to future researchers of building on existing work.

IN-HOUSE REPLICATION

The experience of the *QJPS* in its first decade underscores the importance of its policy of in-house review. Before publication, all replication packages are tested to ensure that code runs cleanly, it is interpretable, and it generates the results in the paper. This process—although it varies by paper—takes approximately six hours of labor and costs roughly $180 per paper; problematic papers occasionally require substantially more time.

This level of review represents a sensible compromise between the two extremes of review. On the one hand, most scholars would agree that an *ideal* replication consists of a talented researcher re-creating a paper from scratch, based solely on the paper's written methodology section. However, undertaking such a replication for every submitted paper would be cost-prohibitive in time and labor, as would having someone perform a line-by-line check of an author's code for errors. On the other hand, direct publication of replication packages without review also is potentially problematic. Experience reveals that many authors submit replication packages that are extremely difficult to interpret or may not even run, thereby defeating the purpose of a replication policy.

Given that the *QJPS* review is relatively basic, however, we might ask whether it is even worth the considerable time that the *QJPS* invests. Experience has shown that the answer is an unambiguous "yes." Of the 24 empirical papers subject to in-house replication review between September 2012 and November 2015,[1] only four packages did not require any modifications. Of the remaining 20 papers, 13 had code that would not execute without errors, eight failed to include code for results that appeared in the paper,[2] and seven failed to include installation directions for software dependencies. Most troubling, however, was that 14 (58%) papers had results that differed from those generated by the author's own code. Some of these issues were relatively small—likely arising from rounding errors during transcription—but, in other cases, they involved incorrectly signed or mislabeled regression coefficients, significant errors in observation counts, and incorrect summary statistics. Frequently, these discrepancies required changes to full columns or tables of results. Moreover, Zachary Peskowitz, who served as the *QJPS* replication assistant from 2010 to 2012, also reported similar levels of replication errors during his tenure. The extent of the issues—which occurred despite authors having been informed that their packages would be subject to review—indicates the necessity for this type of in-house interrogation of code before publication.

Of the 24 empirical papers subject to in-house replication review between September 2012 and November 2015,[1] only four packages did not require any modifications. Of the remaining 20 papers, 13 had code that would not execute without errors, eight failed to include code for results that appeared in the paper,[2] and seven failed to include installation directions for software dependencies.

ADDITIONAL CONSIDERATIONS FOR A REPLICATION POLICY

This section is an overview of the most pressing and concrete considerations that the *QJPS* has come to view as central to a successful replication policy. These considerations—and the specific policies adopted to address them—are the result of hard-learned lessons from a decade of replication experience.

Ease of Replication

The primary goal of *QJPS* policies is to ensure that replication materials can be used and interpreted easily. To the *QJPS*, ease of replication means that those who want to replicate a published article (hereinafter, a "replicator") should be able to do so, as follows:

1. Open a README.txt file in the root replication folder and find a summary of all replication materials in that folder, including any subfolders.
2. After installing any required software and setting a working directory according to directions provided in the README.txt file, open and run the relevant files to generate every result and figure in the publication. This includes all results in print and/or online appendices.
3. Once the code has finished running, easily locate the output and see where that output is reported in the paper's text, footnotes, figures, tables, and appendices.

README.txt File

To facilitate ease of replication, all replication packages should include a README.txt file that includes, at a minimum, the following:

1. *Table of Contents.* A brief description of every file in the replication folder.
2. *Notes for each table and figure.* A short list of where replicators will find the code needed to replicate all parts of the publication.
3. *Base software dependencies.* A list of all software required for replication, including the version of software used by the author (e.g., Stata 11.1, R 2.15.3, 32bit Windows 7, or OSX 10.9.4).
4. *Additional dependencies.* A list of all libraries or added functions required for replication, as well as the library and function versions that were used and the location from which they were obtained.
   a. *R.* The current R versions are found by typing *R.Version()*; information on loaded libraries is found by typing *sessionInfo()*.
   b. *Stata.* Stata does not specifically "load" extra functions in each session; however, a list of all add-ons installed on a system is found by typing *ado dir*.
5. *Seed locations.* Authors are required to set seeds in their code for analyses that use randomness (e.g., simulations or

https://doi.org/10.1017/S1049096516000196 Published online by Cambridge University Press

bootstrapped standard errors). The README.txt file should include a list of locations where seeds are set in the analyses so that replicators can find and change them to check robustness of the results.

Depth of Replication

The *QJPS* requires every replication package to include the code that computes primary results of the paper. That is, it is not sufficient to provide a file of precomputed results along with the code that formats those results for *LaTeX*. Rather, the replication package must include everything necessary to execute the statistical analyses or simulations that constitute the primary contribution of the paper. For example, if the primary contribution is a set of regressions, then the data and code needed to produce those regressions must be included. If the primary contribution is a simulation, then code for that simulation must be provided—not simply a dataset of the simulation results. If the primary contribution is a novel estimator, then code for the estimator must be provided. If the primary contribution is theoretical and numeric simulation or approximation methods were used to provide the equilibrium characterization, then that code must be included.

Although the *QJPS* does not necessarily require the submitted code to access the data if they are publicly available (e.g., data from the National Election Studies or another data repository), it does require that the dataset containing all of the original variables used in the analysis be included in the replication package. In the interest of transparency, the variables should be in their original, untransformed, and unrecoded form, including code that performs the transformations and recodings in the reported analyses. This allows replicators to assess the impact of transformations and recodings on the results.

Proprietary and Non-Public Data

If an analysis relies on proprietary or non-public data, authors are required to contact *QJPS* editors before or during initial submission. Even when data cannot be released publicly, authors often are required to provide *QJPS* staff access to the data for replication before publication. This may require additional arrangements—in the past, it was necessary for *QJPS* staff to be written in Institutional Review Board authorizations. However, in-house review is especially important in these contexts because papers based on non-public data are difficult if not impossible for other scholars to interrogate postpublication.

Software Dependencies

Online software repositories—such as CRAN and SSC—provide authors with easy access to the latest versions of powerful add-ons to standard programs such as R and Stata. Yet, the strength of these repositories—the ability to ensure that authors are always working with the latest version—is also a liability for replication.

Because online repositories always provide the most recent version of add-ons to users, the software provided in response to a given query actually changes over time. Experience has shown that this can cause problems when authors use calls to these repositories to install add-ons through commands such as *install_packages("PACKAGE")* in R or *ssc install PACKAGE* in Stata. Because scholars may attempt to replicate papers months or even years after a paper has been published, changes in the software provided in response to those queries may lead to replication failures. Indeed, the *QJPS* has experienced replication failures due to changes in the software hosted on the CRAN server that occurred between the time when a paper was submitted and when it was reviewed.

Therefore, the *QJPS* now requires authors to include copies of all software (both base software and add-on functions and libraries) used in the replication in their package, as well as code that installs the package on a replicator's computer. The only exceptions are extremely common tools, such as R, Stata, Matlab, Java, Python, and ArcMap (although Java- and Python-based applications must be included).[3]

Randomizations and Simulations

Several modern algorithms use randomness in generating results (e.g., the bootstrap). In these cases, replication requires ensuring that both (a) the *exact* results in the paper can be re-created, and (b) the results in the paper are typical rather than "cherrypicked" outliers. To facilitate this type of analysis, authors should do the following:

Because scholars may attempt to replicate papers months or even years after a paper has been published, changes in the software provided in response to those queries may lead to replication failures.

1. Set a random-number generator seed in their code so that it consistently generates the exact results in the paper.
2. Provide a note in the README.txt file that indicates the location of all of these commands so that replicators can remove them and test the representativeness of the results.

Despite these precautions, painstaking experience has shown that setting a seed is not always sufficient to ensure exact replication. For example, some libraries generate slightly different results on different operating systems (e.g., Windows versus OSX) and on different hardware architectures (e.g., 32-bit versus 64-bit Windows 7). To protect authors, *QJPS* encourages them to test their code on multiple platforms and to document any resulting exceptions or complications in the README.txt file.

ArcGIS

Although *QJPS* encourages authors to write replication code for their ArcGIS-based analyses using the ArcPy scripting utility, it recognizes that most authors have yet to adopt this tool. At this time, the *QJPS* accepts detailed, step-by-step instructions for replicating results via the ArcGIS Graphical User Interface (GUI). However, similar to the inclusion and installation of add-on functions, the *QJPS* has made a tutorial for using ArcPy

https://doi.org/10.1017/S1049096516000196 Published online by Cambridge University Press

available to authors, which is expected to accelerate the transition in using this tool.[4]

ADVICE TO AUTHORS

In addition to the preceding requirements, the *QJPS* provides authors with simple guidelines to prevent common errors. The following suggestions are not mandatory but are highly recommended:

1. *Test files on a different computer, preferably with a different operating system.* After replication code has been prepared, the *QJPS* suggests that authors email it to a different computer, unzip it, and run it. Code often contains small dependencies—such as software requirements or specific file locations—that remain unnoticed until replication. Running code on a different computer often exposes these issues in a way that running the code on an author's own computer does not.
2. *Check every code-generated result against the final manuscript PDF.* The majority of replication problems emerge because authors either modified their code but failed to update their manuscript or made an error while transcribing their results into their paper. Therefore, authors are strongly encouraged to print out a copy of their manuscript and check each result before submitting the final version and the replication package.

CONCLUSION

As the nature of academic research changes, becoming ever more computationally intense, so too must the peer-review process evolve. This article is an overview of many of the lessons learned by the attempt of *QJPS* to address this need. Most important, however, this article documents not only the importance of requiring the transparent publication of replication materials but also the strong need for in-house review of these materials before publication. ■

As the nature of academic research changes, becoming ever more computationally intense, so too must the peer-review process evolve.

NOTES

1. This is the period during which the author was responsible for all in-house interrogations of replication packages at the *QJPS*.
2. This does not include code that failed to execute, which also might be perceived as failing to replicate results from the paper.
3. To aid researchers in meeting this requirement, detailed instructions on how to include CRAN and SSC packages in replication packages are provided by the *QJPS*.
4. ArcPy is a Python-based tool for scripting in ArcGIS.

https://doi.org/10.1017/S1049096516000196 Published online by Cambridge University Press